



Performance (QoS) Optimization

Multiple QoS Levels



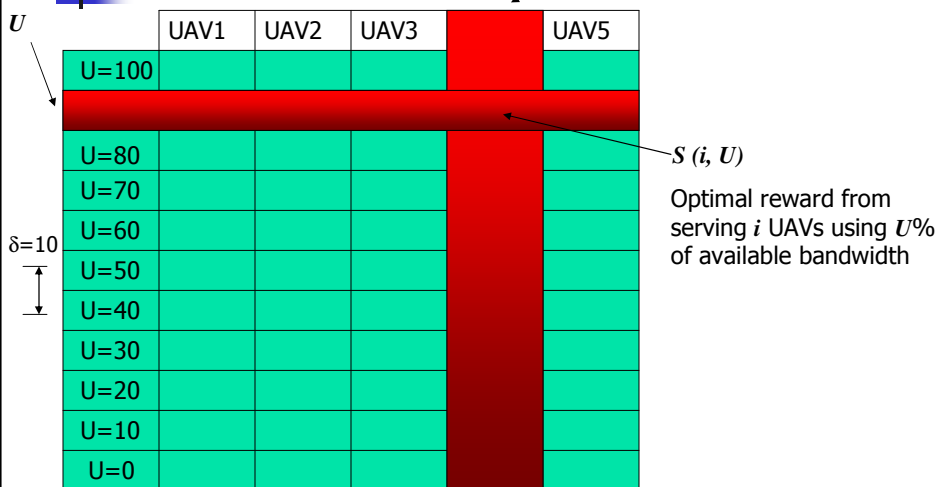
QoS Adaptation

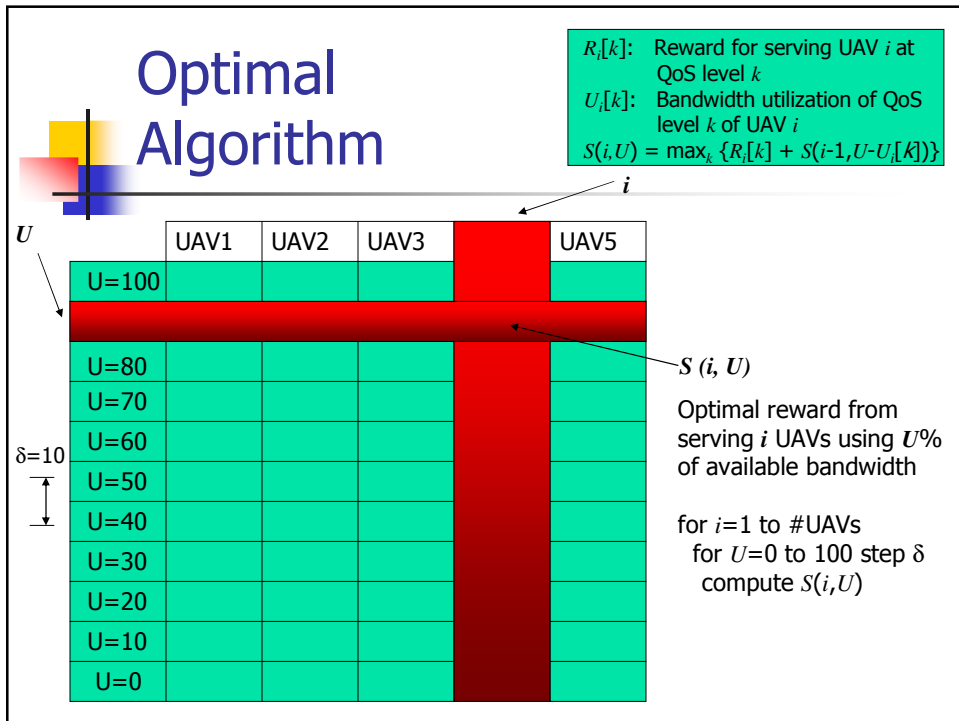
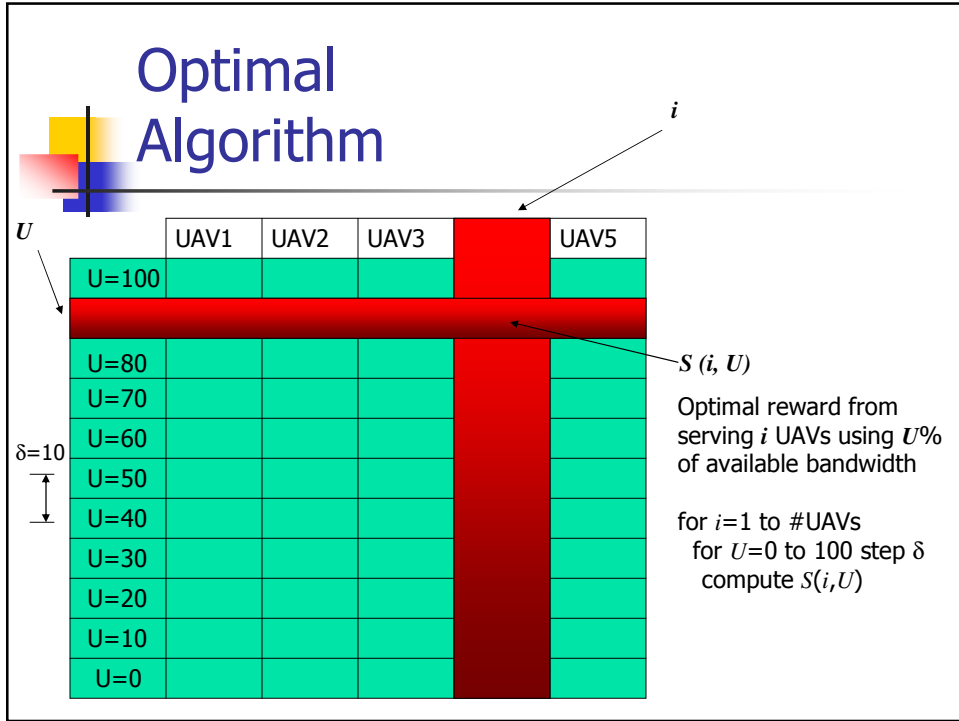
- A set of UAVs send video streams and audio streams with multiple of QoS levels to a base station
 - UAV picture can be transmitted at three resolutions:
 - Level1: 40% Avail. Bandwidth, 100% Utility
 - Level2: 30% Avail. Bandwidth, 70% Utility
 - Level3: 10% Avail. Bandwidth, 10% Utility
 - UAV audio can be transmitted at three resolutions:
 - Level1: 30% Avail. Bandwidth, 60% Utility
 - Level2: 20% Avail. Bandwidth, 50% Utility
 - Level3: 10% Avail. Bandwidth, 40% Utility
- How to maximize system utility?

QoS Optimization

	UAV1	UAV2	UAV3	UAV4	UAV5
U=100					
U=90					
U=80					
U=70					
U=60					
U=50					
U=40					
U=30					
U=20					
U=10					
U=0					

QoS Optimization





QoS Optimization Example

Example

- Client 1, 2:
 - Level1: 40% CPU, \$1
 - Level2: 30% CPU, \$0.7
 - Level3: 10% CPU, \$0.1
- Client 3, 4:
 - Level1: 30% CPU, \$0.65
 - Level2: 20% CPU, \$0.6
 - Level3: 10% CPU, \$0.4

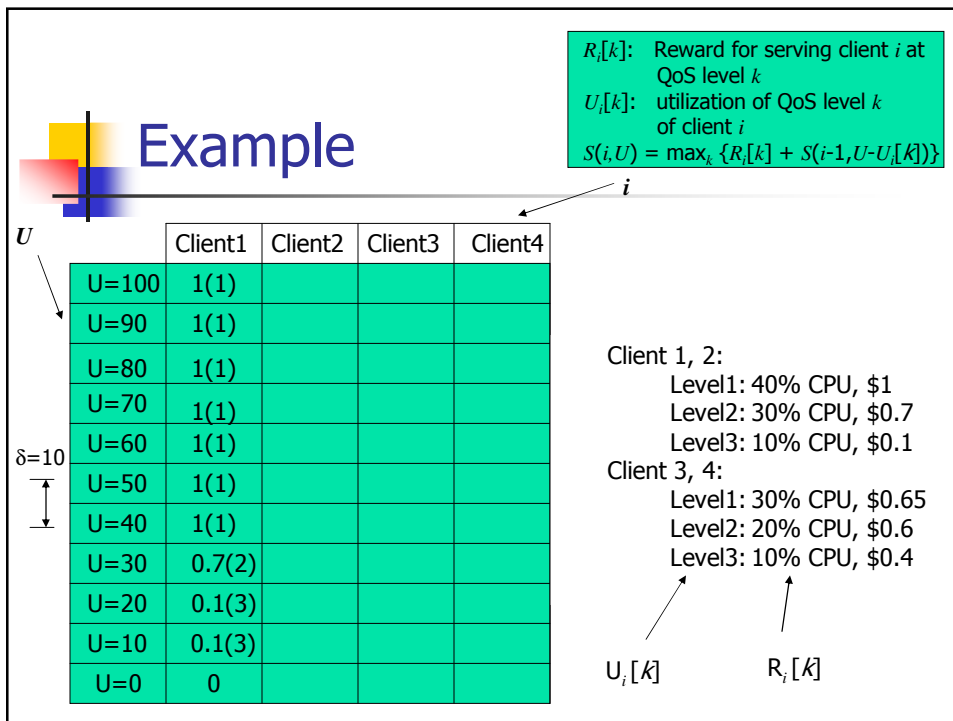
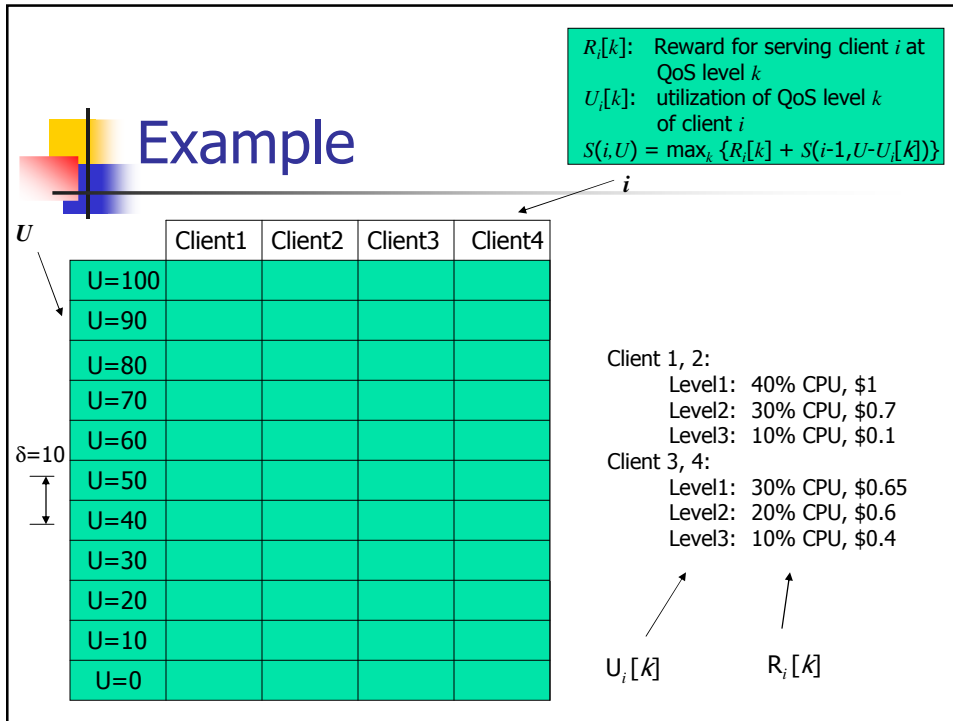
QoS Optimization Example (Client Tasks on a Server)

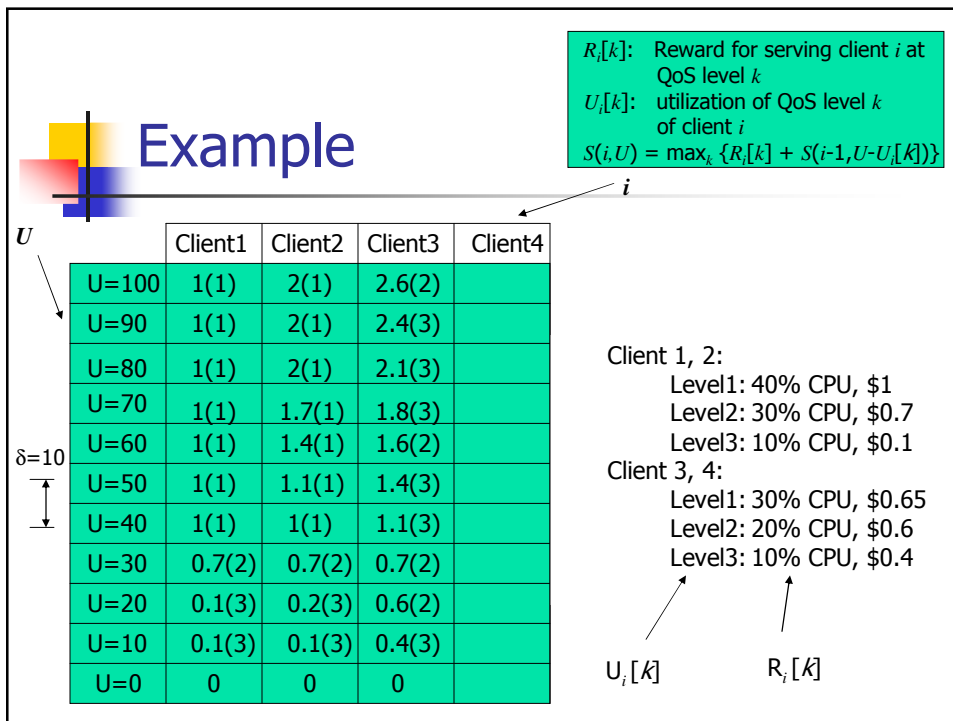
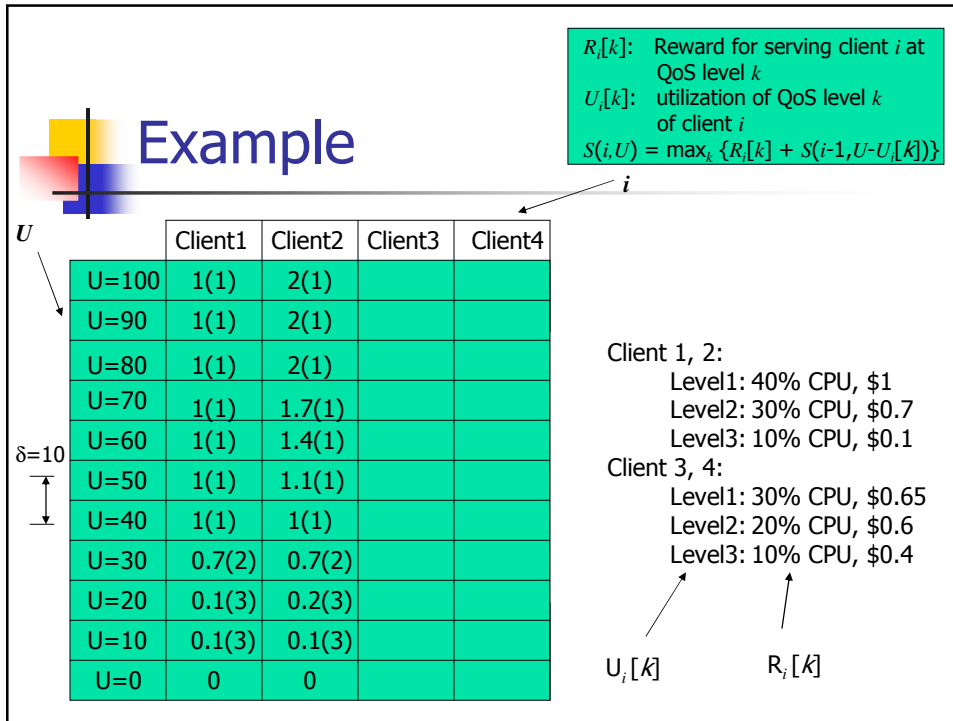
Example ^{*i*}

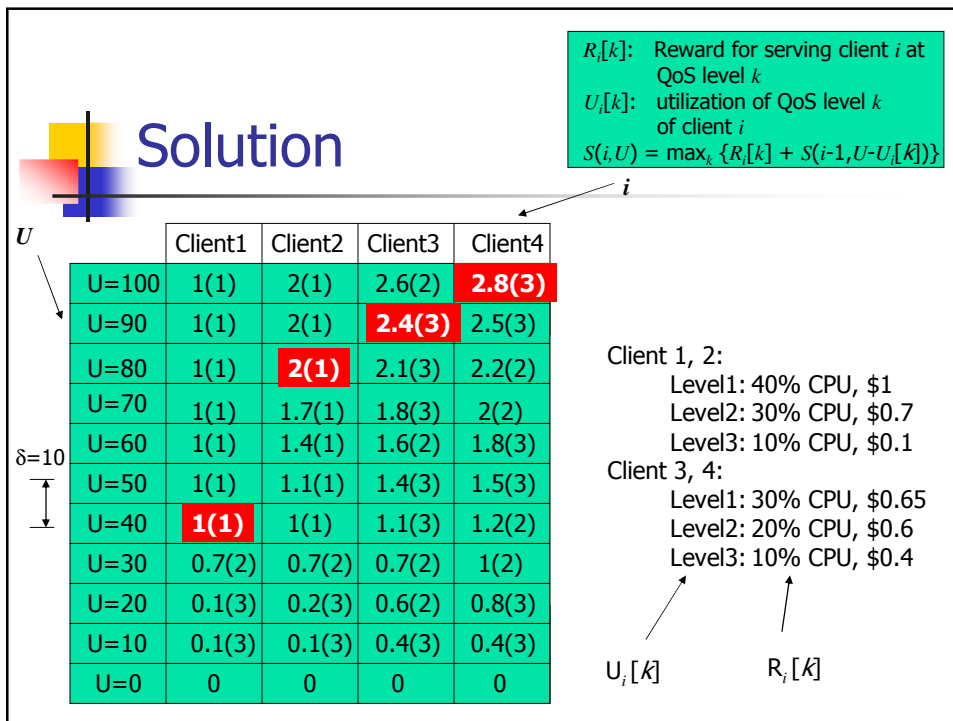
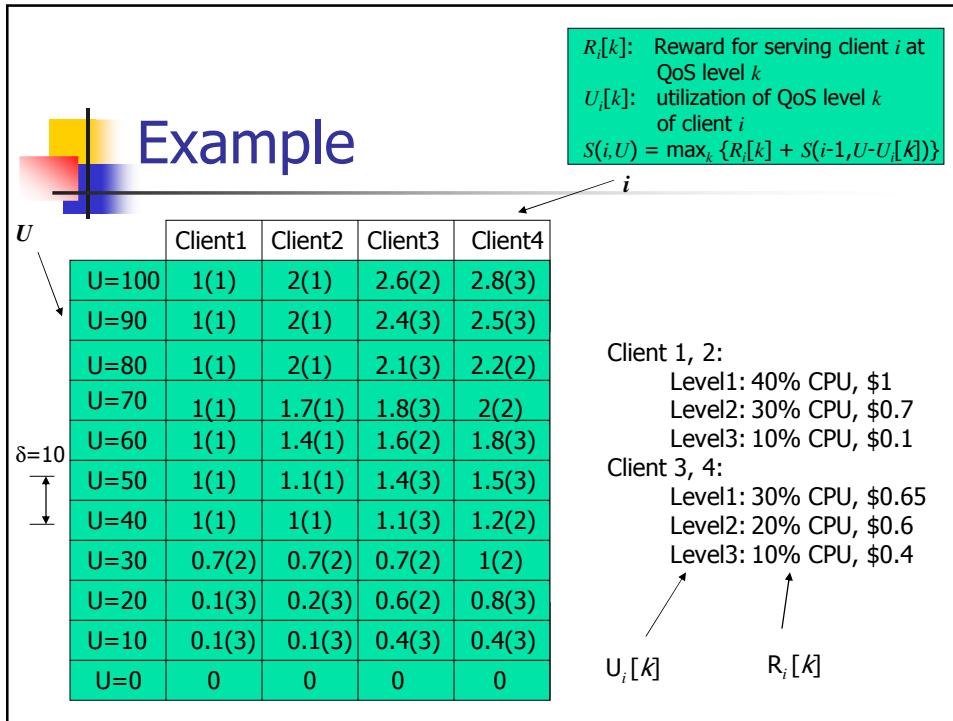
- Client 1, 2:
 - Level1: 40% CPU, \$1
 - Level2: 30% CPU, \$0.7
 - Level3: 10% CPU, \$0.1
- Client 3, 4:
 - Level1: 30% CPU, \$0.65
 - Level2: 20% CPU, \$0.6
 - Level3: 10% CPU, \$0.4

$U_i[k]$ ↗

↖ $R_i[k]$







Approximate QoS Optimization

Hill Climbing Algorithm

How to get quickly to the top and stay there longest?



Approximate QoS Optimization

Hill Climbing Algorithm

How to get quickly to the top and stay there longest?

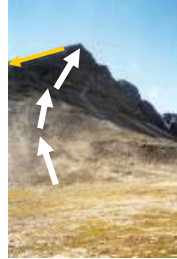


- Ascend the path of maximum slope
- Descend the path of minimum slope

Approximate QoS Optimization

Hill Climbing Algorithm

How to get quickly to the top and stay there longest?



- Ascend the path of maximum slope
- Descend the path of minimum slope

Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm

How to get quickly to the top and stay there longest?



- Ascend the path of maximum slope
- Descend the path of minimum slope

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Loop
 if underutilized then take maximum **slope** promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

0% 0,0,0,0

Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Promote 3 or 4 to L3: Slope=0.4/0.1=4
 Promote 1 or 2 to L1: Slope=1/0.4=2.5
 Promote 3 or 4 to L2: Slope=0.6/0.2=3
 Promote 1 or 2 to L2: Slope=0.7/0.3=2.33
 Promote 3 or 4 to L1: Slope=0.65/0.3=2.02
 Promote 1 or 2 to L3: Slope=0.1/0.1=1

0% 0,0,0,0

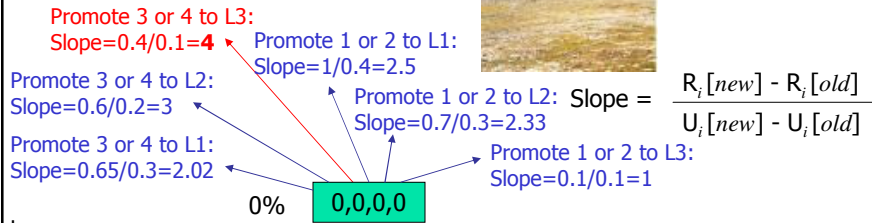
Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4



Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4



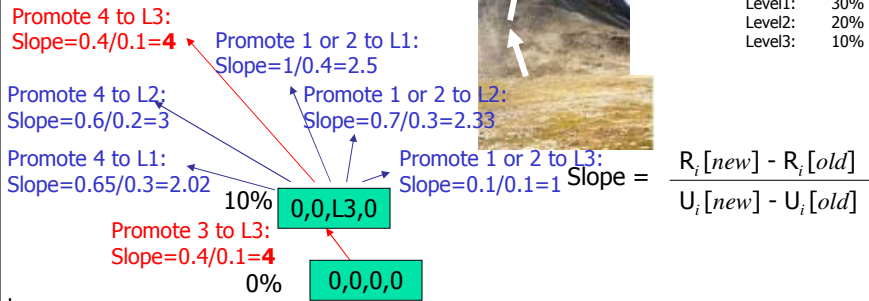
Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4



$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

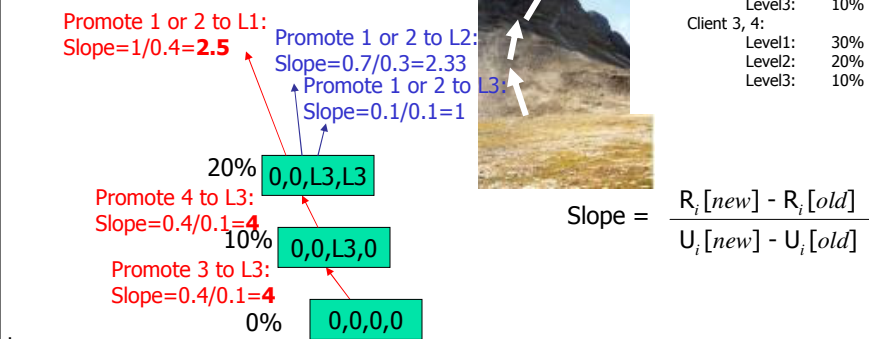
Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

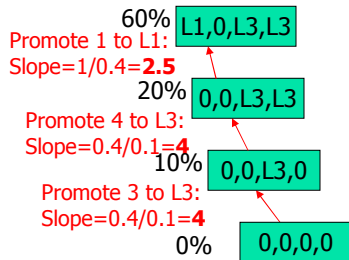


$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



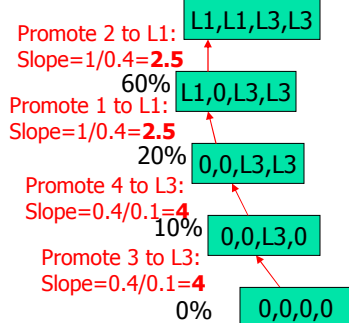
$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



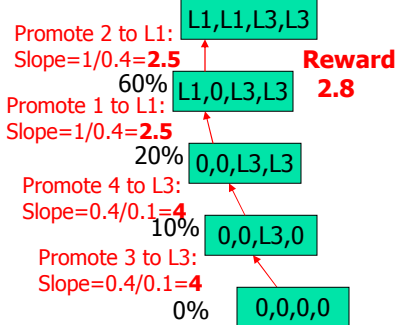
$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1
 Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

Loop
 if underutilized then take maximum slope promotion
 if overload then take minimum slope demotion
 End Loop

Approximate QoS Optimization

Hill Climbing Algorithm



$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:

Level1:	40% CPU, \$1
Level2:	30% CPU, \$0.7
Level3:	10% CPU, \$0.1

Client 3, 4:

Level1:	30% CPU, \$0.65
Level2:	20% CPU, \$0.6
Level3:	10% CPU, \$0.4

Loop
if underutilized then take maximum slope promotion
if overload then take minimum slope demotion
End Loop

Approximate QoS Optimization

Restricted Hill Climbing



$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:

Level1:	40% CPU, \$1
Level2:	30% CPU, \$0.7
Level3:	10% CPU, \$0.1

Client 3, 4:

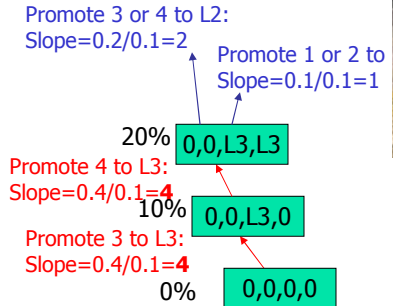
Level1:	30% CPU, \$0.65
Level2:	20% CPU, \$0.6
Level3:	10% CPU, \$0.4

0% **0,0,0,0**

Loop
if underutilized then take maximum slope *single level* promotion
if overload then take minimum slope *single level* demotion
End Loop

Approximate QoS Optimization

Restricted Hill Climbing



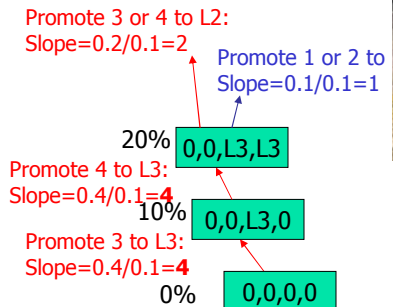
$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:	
Level1:	40% CPU, \$1
Level2:	30% CPU, \$0.7
Level3:	10% CPU, \$0.1
Client 3, 4:	
Level1:	30% CPU, \$0.65
Level2:	20% CPU, \$0.6
Level3:	10% CPU, \$0.4

Loop
 if underutilized then take maximum slope *single level* promotion
 if overload then take minimum slope *single level* demotion
 End Loop

Approximate QoS Optimization

Restricted Hill Climbing



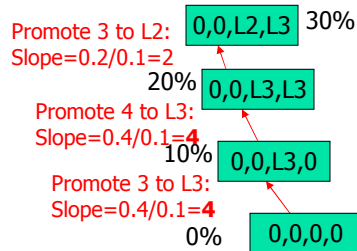
$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:	
Level1:	40% CPU, \$1
Level2:	30% CPU, \$0.7
Level3:	10% CPU, \$0.1
Client 3, 4:	
Level1:	30% CPU, \$0.65
Level2:	20% CPU, \$0.6
Level3:	10% CPU, \$0.4

Loop
 if underutilized then take maximum slope *single level* promotion
 if overload then take minimum slope *single level* demotion
 End Loop

Approximate QoS Optimization

Restricted Hill Climbing



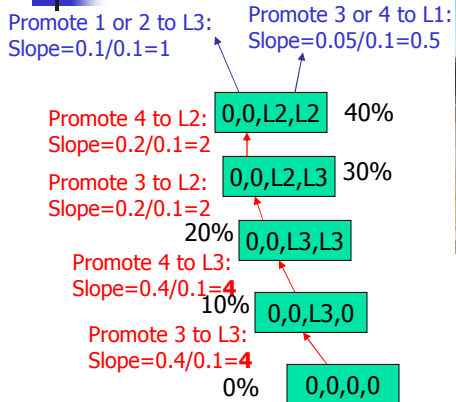
$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:
Level1: 40% CPU, \$1
Level2: 30% CPU, \$0.7
Level3: 10% CPU, \$0.1

Client 3, 4:
Level1: 30% CPU, \$0.65
Level2: 20% CPU, \$0.6
Level3: 10% CPU, \$0.4

Loop
if underutilized then take maximum slope *single level* promotion
if overload then take minimum slope *single level* demotion
End Loop

Approximate QoS Optimization



$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Client 1, 2:
Level1: 40% CPU, \$1
Level2: 30% CPU, \$0.7
Level3: 10% CPU, \$0.1

Client 3, 4:
Level1: 30% CPU, \$0.65
Level2: 20% CPU, \$0.6
Level3: 10% CPU, \$0.4

Loop
if underutilized then take maximum slope *single level* promotion
if overload then take minimum slope *single level* demotion
End Loop

Approximate QoS Optimization

Promote 1 or 2 to L3:
Slope=0.1/0.1=1


Promote 3 or 4 to L1:
Slope=0.05/0.1=0.5

Promote 4 to L2:
Slope=0.2/0.1=2

Promote 3 to L2:
Slope=0.2/0.1=2

Promote 4 to L3:
Slope=0.4/0.1=4

Promote 3 to L3:
Slope=0.4/0.1=4



Client 1, 2:
Level1: 40% CPU, \$1
Level2: 30% CPU, \$0.7
Level3: 10% CPU, \$0.1

Client 3, 4:
Level1: 30% CPU, \$0.65
Level2: 20% CPU, \$0.6
Level3: 10% CPU, \$0.4

0% 0,0,0,0

10% 0,0,L3,0

20% 0,0,L3,L3


30% 0,0,L2,L3

40% 0,0,L2,L2

Slope =
$$\frac{R_i[new] - R_i[old]}{U_i[new] - U_i[old]}$$

Loop
if underutilized then take maximum slope *single level* promotion
if overload then take minimum slope *single level* demotion
End Loop

Approximate QoS Optimization



Client 1, 2:
Level1: 40% CPU, \$1
Level2: 30% CPU, \$0.7
Level3: 10% CPU, \$0.1

Client 3, 4:
Level1: 30% CPU, \$0.65
Level2: 20% CPU, \$0.6
Level3: 10% CPU, \$0.4

50% L3,0,L2,L2

Slope =
$$\frac{R_i[new] - R_i[old]}{U_i[new] - U_i[old]}$$

Loop
if underutilized then take maximum slope *single level* promotion
if overload then take minimum slope *single level* demotion
End Loop



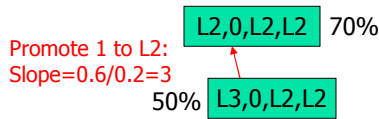
Approximate QoS Optimization



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1

Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$



Loop
 if underutilized then take maximum slope *single level* promotion
 if overload then take minimum slope *single level* demotion
 End Loop



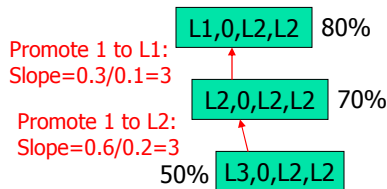
Approximate QoS Optimization



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1


Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$



Loop
 if underutilized then take maximum slope *single level* promotion
 if overload then take minimum slope *single level* demotion
 End Loop

Approximate QoS Optimization



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1

Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

Promote 2 to L3:
 Slope=0.1/0.1=1

Promote 1 to L1:
 Slope=0.3/0.1=3

Promote 1 to L2:
 Slope=0.6/0.2=3

50% L3,0,L2,L2


70% L2,0,L2,L2

80% L1,0,L2,L2

90% L1,L3,L2,L2

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Loop
 if underutilized then take maximum slope *single level* promotion
 if overload then take minimum slope *single level* demotion
 End Loop



Client 1, 2:
 Level1: 40% CPU, \$1
 Level2: 30% CPU, \$0.7
 Level3: 10% CPU, \$0.1

Client 3, 4:
 Level1: 30% CPU, \$0.65
 Level2: 20% CPU, \$0.6
 Level3: 10% CPU, \$0.4

100% L1,L3,L1,L2

Promote 3 to L1:
 Slope=0.05/0.1=0.5

Promote 1 to L1:
 Slope=0.1/0.1=1

Promote 1 to L2:
 Slope=0.3/0.1=3

Promote 2 to L3:
 Slope=0.6/0.2=3

50% L3,0,L2,L2

70% L2,0,L2,L2

80% L1,0,L2,L2

90% L1,L3,L2,L2

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Loop
 if underutilized then take maximum slope *single level* promotion
 if overload then take minimum slope *single level* demotion
 End Loop



Approximate QoS Optimization



Client 1, 2:

Level1:	40% CPU, \$1
Level2:	30% CPU, \$0.7
Level3:	10% CPU, \$0.1

Client 3, 4:

Level1:	30% CPU, \$0.65
Level2:	20% CPU, \$0.6
Level3:	10% CPU, \$0.4

$$\text{Slope} = \frac{R_i[\text{new}] - R_i[\text{old}]}{U_i[\text{new}] - U_i[\text{old}]}$$

Loop
 if underutilized then take maximum slope *single level* promotion
 if overload then take minimum slope *single level* demotion
 End Loop



QoS Optimization

Multiple QoS Levels
 Multiple Resources



QoS Adaptation in Multiple Dimensions

- QoS Dimensions (example)
 - Video resolution
 - Color depth
 - Frame rate
 - Cryptographic security
 - Audio quality
- Quality space Q_i for task T_i
 - (possible video resolutions, possible color depths, possible frame rates, possible security policies, possible audio encoding policies)



QoS Adaptation in Multiple Dimensions

- Element q_i of Quality space Q_i for task T_i is a particular allocation of QoS choices, for example:
 - $q_i =$ (resolution = 800x600,
color depth = 24 bits,
frame rate = 30 frames/sec,
security policy = 128 bit key encryption,
audio quality = 44 kbps)

QoS Adaptation in Multiple Dimensions

- Dimension j of the Quality space Q_i for task T_i is an ordered set Q_{ij} of values $\{q_{i1}, \dots, q_{id}\}$, for example:

- Video resolution

- $Q_{ij} = (640 \times 480, 800 \times 600, 1024 \times 768)$

$$q_{ij} > q_{ijmin}$$

$$u_i = \sum_{j=1}^{d_i} w_{ij} u_{ij}$$

Utility u_{i1} u_{i2} u_{i3}

Maximize u subject to quality and resource constraints

$$u = \sum_{j=1}^n w_i u_i$$

Resource Profile

- $g_i(r)$: Given an amount of resources r_1, \dots, r_m what is the maximum achievable utility for task i ? (think "first column in the dynamic programming table")
- $h_i(r)$: Is the corresponding QoS level allocation

Recursive step (case of two resources):

$$v(i, p_1, p_2) = \max_{p'_1, p'_2} \{g_i(p'_1, p'_2) + v(i-1, p_1 - p'_1, p_2 - p'_2)\}$$



Task Utility/Resource List

Let $C_i = (u_{i1}, r_{i1}), \dots, (u_{ik}, r_{ik})$

list the discontinuity points in g_i .

Utility/Penalty vector:

$C_i = (u_{i1}, r_{i1}, r_{i1}^*), \dots, (u_{ik}, r_{ik}, r_{ik}^*)$

r^* is a scalar "cost" summarizing vector resource consumption

Convex Hull \rightarrow greedy hill climbing